

УНІВЕРСИТЕТ «КРОК»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТА КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

МЕТОДИЧНІ НАСТАНОВИ
до виконання здобувачами вищої освіти
Проєкту третього рівня

ЗАВТЕРДЖЕНО
на засіданні кафедри
комп'ютерних наук
протокол № 8 від 21.04.2022

Київ, 2022

Методичні настанови до виконання здобувачами вищої освіти Проєкту третього рівня / уклад. Єпик О.М., Тимчук О.С.; Київ, Університет «КРОК». 2022. 40 с.

Укладачі:

Єпик М.О., канд. техн. наук; Тимчук О.С., канд. техн. наук.

Рецензенти:

Пилипенко А.І., канд. техн. наук

Барибін О.І., канд. техн. наук

Проєкт третього рівня виконується здобувачами спеціальності 122 «Комп'ютерні науки» у відповідності до навчального плану 3 курсу у 6 семестрі.

*Метою виконання Проєкту є проектування та реалізація **програмного забезпечення з базою даних**, логікою доступу до даних, логікою програми та логікою інтерфейсу.*

База даних розробляється на основі реляційної моделі даних. При цьому нормалізація має бути виконана до рівня вимог третьої нормальної форми (ЗНФ).

Виконання Проєкту виконується відповідно до рекомендацій щодо створення баз даних, що наведені у курсі *Database Foundations* від *Oracle Academy*.

Доступ здобувачам до методичних рекомендацій, програмного забезпечення та хмарних сервісів надається відповідно до угоди про членство університету «КРОК» у *Oracle Academy*.

ПЕРЕДМОВА

Положення методичних настанов рекомендовані до застосовування під час навчання здобувачами освітнього ступеня «Бакалавр» на освітній програмі за спеціальністю 122 «Комп'ютерні науки» відповідно до графіку навчального процесу.

Текст методичних настанов містить положення, які розкривають рекомендовані дії здобувачів під час виконання Проєкту.

Для більш глибокого розуміння положень методичних настанов та з'ясування додаткової довідкової інформації здобувачам рекомендовано ознайомитись з текстами релевантних документів, які регламентують організацію освітнього процесу в Університеті «КРОК», регулюють відносини, що виникають між його учасниками, є невід'ємною складовою системи забезпечення якості освітньої діяльності та якості вищої освіти (системи внутрішнього забезпечення якості) Університету.

До таких основних документів належать:

- 1) Положення про організацію освітнього процесу в Університеті «КРОК» [1];
- 2) Положення про академічну доброчесність в Університеті «КРОК» [2];
- 3) Положення про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» [3].

Для якісного написання пояснювальної записки, підготовки до презентації та захисту рекомендовано також ознайомитись з текстами державних стандартів, які встановлюють вимоги до оформлення наукових текстів та бібліографічного опису використаних джерел:

- 1) ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання [4];
- 2) ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання [5].

ЗМІСТ

| | |
|---|----|
| ВСТУП | 5 |
| 1 ТЕОРЕТИЧНІ ВІДОМОСТІ | 7 |
| 1.1 Scrum-методологія | 7 |
| 1.2 Визначення і структура бази даних | 13 |
| 1.3 Тестування | 18 |
| 2 СТРУКТУРА ПРОЄКТУ | 25 |
| 3 ПОРЯДОК ВИКОНАННЯ ПРОЄКТУ | 26 |
| 2.1 Етапи виконання Проєкту | 26 |
| 2.2 Реліз програмного забезпечення | 26 |
| 2.3 Структура та вимоги до пояснювальної записки | 27 |
| 2.4 Структура та вимоги до презентації | 30 |
| 4 КРИТЕРІЇ ОЦІНЮВАННЯ ПРОЄКТУ | 33 |
| 5 ДОТРИМАННЯ ВИМОГ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ПРИ ВИКОНАННІ ПРОЄКТУ | 34 |
| 6 ТЕХНІЧНЕ ОБЛАДНАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ | 38 |
| 7 ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ | 39 |
| ДОДАТОК А | 40 |

ВСТУП

Проект третього рівня здобувач вищої освіти освітньо-професійної програми за спеціальністю 122 «Комп'ютерні науки» [6] реалізує у шостому семестрі.

Робота над Проектом передбачає встановлення відповідності засвоєних здобувачами рівня й обсягу знань, умінь, інших компетентностей до вимог стандарту вищої освіти першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп'ютерні науки» (затверджено та введено в дію наказом Міністерства освіти і науки України від 10.07.2019 р. № 962) [6].

У ході виконання Проекту здобувачі мають продемонструвати свою здатність:

- 1) до системного мислення, застосування методології системного аналізу для дослідження складних проблем різної природи, методів формалізації та розв'язування системних задач, що мають суперечливі цілі, невизначеності та ризику;
- 2) проєктувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління;
- 3) застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника;
- 4) до аналізу предметних областей різних галузей знань, збору детальної інформації для формалізації функціональних та нефункціональних вимог до програмного забезпечення, ідентифікації, класифікації та пошуку методів і підходів щодо їх розв'язання;
- 5) до командної роботи при розробці програмного забезпечення протягом його життєвого циклу (від збору вимог до впровадження) та різних середовищ;
- 6) до написання дизайн-документів з використанням UML-нотації;
- 7) проводити тестування компонентів програмного забезпечення.

У результаті роботи над Проектом здобувачі вищої освіти мають навчитися:

- 1) проєктувати концептуальні і логічні моделі баз даних та створювати фізичні моделі бази даних;
- 2) розробляти реляційні бази даних;
- 3) створювати SQL-запити до баз даних;
- 4) працювати в команді, проводити групові дослідження і вирішувати задачі із застосуванням відповідних теорій та методів інформаційних технологій;
- 5) розробляти архітектури, модулі та компоненти складних інформаційних систем;
- 6) обґрунтовано вибирати та використовувати патерни та фреймворки проєктування і програмування для вирішення типових підзадач при розробці програмного забезпечення;
- 7) мотивовано обирати мови програмування та технології для розв'язання завдань створення і супроводження інформаційних систем відповідно до специфікацій і потреб користувача.

Написання, підготовка до захисту та захист Проекту базується на таких основних принципах, як усвідомлення важливості академічної доброчесності та відповідальності за її порушення, а також нульової толерантності до порушення академічної доброчесності [2].

Термін виконання та захисту Проекту визначається графіком освітнього процесу.

Успішно захищені Проекти здобувачів оприлюднюються на офіційному сайті (або у репозиторії) університету.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Scrum-методологія

Методологія розробки програмного забезпечення (ПЗ) – це шаблон, що визначає взаємодію різних складових процесу розробки ПЗ. Важливо обрати методологію, яка буде найбільш відповідною для проєкту, що дозволить суттєво знизити витрати та підвищити якість результату.

Існують наступні види методологій розробки ПЗ:

- 1) *каскадна або поетапна розробка*: модель «Водоспад» (*Waterfall*), каскадна модель з проміжним контролем, V-образна модель (розробка через тестування) – процес створення ПЗ, який є потоком, що послідовно проходить фази аналізу, проєктування, реалізацій, тестування, інтеграції та підтримки (зазвичай використовується для розробки крупних проєктів з тривалим терміном впровадження, на даний час майже не використовується);
- 2) *ітеративна інкрементна* (еволюційна) модель: дана методологія дозволяє паралельно виконувати ряд завдань з безперервним аналізом результатів і коректуванням попередніх етапів роботи (робота для великого штату кваліфікованих програмістів): RUP (*Rational Unified Process*);
- 3) *спіральна методика* (*Spiral*) характеризується проходженням проєктом циклу, що повторюється в кожній фазі розвитку: планування-реалізація-перевірка-оцінка (*plan-do-check-act cycle*) (використовується для створення проєктів, які не мають остаточно сформованого результату або вимагають термінового впровадження по етапах);
- 4) *гнучкі технології або Agile-розробка*: Scrum, XP (*Extreme Programming*), Crystal-методології, DSDM (*Dynamic Systems Development Model*), FDD (*Feature Driven Development*) – швидка розробка без шкоди якості, коли головним є працюючий продукт, а не його документація; сучасний неформалізований підхід до створення ПЗ в процесі якого реагування на зміни цінується вище строго дотримання плану (для проєктів, що швидко розвиваються і з кожною ітерацією ПЗ готови до його релізу);
- 5) *модель, що орієнтована на користувача*: JAD-модель (*Joint Application Development*);
- 6) *модель швидкої розробки*: RAD (*Rapid Application Development*).

Гнучка методологія розробки або *Agile-методологія* – серія підходів до розробки ПЗ, що орієнтовані на використання інтерактивної розробки, динамічного формування вимог і забезпечення їх реалізації у результаті

постійної взаємодії всередині робочих груп, що самоорганізуються і складаються із спеціалістів різного профілю. Існує декілька методик, що відносяться до класу гнучких методологій розробки, а саме Scrum, XP (*Extreme Programming*), Crystal-методології, DSDM (*Dynamic Systems Development Model*), FDD (*Feature Driven Development*).

Більшість гнучких методологій націлено на мінімізацію ризиків шляхом зведення розробки всього проєкту до серії стислих циклів, що є ітераціями, які зазвичай тривають 2-3 тижні. Кожна ітерація є програмним проєктом в мініатюрі і включає усі завдання, що необхідні для видачі міні-приросту по функціональності: планування, аналіз вимог, проєктування, програмування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі, що гнучкий програмний проєкт готовий до випуску в кінці кожної ітерації. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розробки.

Практично всі Agile-команди сконцентровані в одному офісі (*bullpen*). Офіс включає *Product owner* – замовника, який і визначає вимоги до продукту. В якості замовника може виступати бізнес-аналітик, менеджер проєкту або клієнт. Крім того, в офіс можуть входити також дизайнери інтерфейсу, тестувальники, технічні письменники. Тобто Agile-методи націлені в першу чергу на безпосереднє спілкування.

Основні ідеї Agile-методології:

- 1) люди і взаємодія важливіше процесів і інструментів;
- 2) працюючий продукт важливіше вичерпної документації;
- 3) співпраця із замовником важливіша за узгодження умов контракту;
- 4) готовність до змін важливіше дотримання початкового плану.

Принципи Agile-методології:

- 1) задоволення клієнта за рахунок раннього і безперебійного постачання цінного програмного забезпечення;
- 2) вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);
- 3) часте постачання робочого програмного забезпечення (кожного місяця або тижня або ще частіше);
- 4) тісне, щоденне спілкування замовника з розробниками впродовж всього проєкту;
- 5) проєктом займаються мотивовані особи, які забезпечені потрібними умовами роботи, підтримкою і довірою;

- 6) метод передачі інформації, що рекомендується, – особиста розмова (обличчям до обличчя);
- 7) працююче програмне забезпечення – кращий вимірник прогресу;
- 8) спонсори, розробники і користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
- 9) постійна увага поліпшенню технічної майстерності і зручному дизайну;
- 10) простота – мистецтво не робити зайвої роботи;
- 11) кращі технічні вимоги, дизайн і архітектура виходять в команди, що самоорганізуються;
- 12) постійна адаптація до обставин, що змінюються.

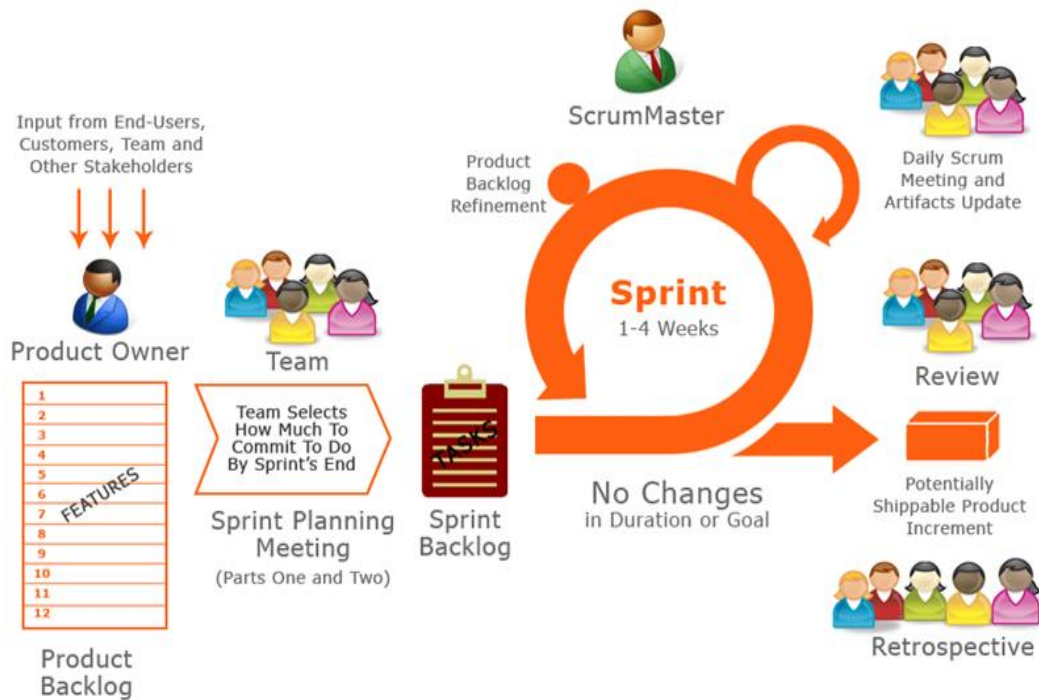
Переваги Agile-методології:

- 1) якість продукту (залучення замовника до процесу кожної ітерації дає можливість коректувати процес, що незмінно підвищує якість);
- 2) висока швидкість розробки (ітерація триває не більш 3-х тижнів, до кінця цього терміну обов'язково є результат);
- 3) мінімізація ризиків (крупний проект дає можливість замовникові сплатити декілька ітерацій і в ході роботи зрозуміти, що він вчасно отримає саме те, що хоче і за прийнятну ціну; замовник завжди має можливість спостерігати за ходом розробки, коректувати функціональність проекту, тестувати або запускати його, навіть може зупинити його у будь-яку мить).

Scrum – це гнучкий метод керування проектами, метою якого є підвищення продуктивності праці в командах розробників. Дана методологія також використовується коли є недолік в наявності досвідчених фахівців на ключових позиціях або чітко сформульованих вимог до продукту.

Своєму терміну «Scrum» зобов'язаний регбі, в якому це слово означає метод командної гри у вигляді побудови трьох ліній кожним з суперників і спроби захопити м'яч. Для успішного перехоплення потрібна не лише хороша фізична підготовка, але і злагодженість кожного учасника сутички і чітке розуміння мети.

Scrum – це методологія керування проектами, яка побудована на принципах тайм-менеджменту. Основною її особливістю є залученість до процесу усіх учасників, кожен з яких має певну роль. Над розв'язанням завдання працює не лише команда розробників, але і всі ті, кого цікавить вирішення даного завдання [11]:



Основні терміни Scrum-методологій:

- 1) **Власник продукту (Product Owner)** – людина, яка має безпосередній інтерес у якісному кінцевому продукті, вона розуміє як цей продукт повинен виглядати та працювати; ця людина не працює у команді розробників, вона працює на боці замовника або клієнта (наприклад, інша компанія, інший відділ і т. д.), але вона є членом команди; дана людина розставляє пріоритети для завдань;
- 2) **Scrum-майстер (Scrum Master)** – керівник проекту, який стежить за дотриманням всіх принципів;
- 3) **Scrum-команда (Scrum Team)** – команда, яка приймає усі принципи **Scrum** і готова з ними працювати;
- 4) **Спринт (Sprint)** – відрізок часу, що береться для виконання визначеного (обмеженого) списку завдань (рекомендований термін – 2-4 тижні, тривалість визначається командою один раз);
- 5) **Беклог (Backlog)** – список усіх робіт (щоденник загального користування); поділяється на 2 види: Product-беклог (повний список усіх робіт, використовуючи які отримуємо кінцевий продукт) та Спринт-беклог (список робіт, які визначила команда і узгодила з Власником продукту на найближчий звітний період (спринт), завдання беруться із Product-беклог);
- 6) **Планування спринту (Sprint Planning Meeting)** – це нарада, де присутні усі (команда, Scrum-майстер, Власник продукту); впродовж цієї наради Власник продукту визначає пріоритетні завдання, які він би хотів побачити виконаними після закінчення терміну; команда оцінює час,

який знадобиться для виконання поставлених завдань; Результатом наради є список вимог, який не може змінюватися протягом спринту і до кінця спринту повинен бути виконаний.

Життєвий цикл спринту складається з:

- 1) планування (*Sprint Planning*);
- 2) зупинки (*Sprint Abnormal Termination*);
- 3) щоденна зустріч (*Daily Scrum Meeting*);
- 4) ретроспектива (*Retrospective*).

Планування спринту (*Sprint Planning*): на початку кожного спринту проводиться планування спринту. У плануванні беруть участь замовники, користувачі, менеджмент, Власник продукту, Scrum-майстер і Scrum-команда. Планування спринту складається з 2 послідовних зустрічей.

У першій зустрічі приймають участь замовники, користувачі, менеджмент, Власник продукту, Scrum-майстер і Scrum-команда. Мета першої зустрічі: визначити мету (*Sprint Goal*) і *Sprint Backlog* – функціональність, яка буде розроблена протягом наступного спринту для досягнення мети спринту. Артефакт (результат зустрічі): *Sprint Backlog*.

У другій зустрічі приймають участь Scrum-майстер і Scrum-команда. Мета другої зустрічі: визначити, як саме буде розроблятися визначена функціональність для того, щоб досягнути мети спринту. Для кожного елемента *Sprint Backlog* визначається список завдань і оцінюється їхня тривалість. Артефакт (результат зустрічі): у *Sprint Backlog* з'являються завдання.

Якщо протягом спринту виявляється, що команда не може встигнути зробити заплановані на спринт завдання, то Scrum-майстер, Власник продукту і Scrum-команда знову зустрічаються і з'ясовують, як можна скоротити обсяг робіт і при цьому досягнути мети спринту.

Зупинка спринту (*Sprint Abnormal Termination*) проводиться у виняткових ситуаціях. Спринт може бути зупинений до того, як закінчатся відведені 30 днів. Спринт може зупинити команда, якщо вона розуміє, що не може досягнути мети спринту в зазначені терміни. Спринт може зупинити Власник продукту, якщо необхідність у досягненні мети спринту зникла. Після зупинки спринту проводиться зустріч з командою, де обговорюються причини зупинки спринту. Після цього починається новий спринт: проводиться його планування і починаються роботи.

Щоденна зустріч (*Daily Scrum Meeting*) проходить кожний ранок на початку дня. Він призначений для того, щоб всі члени команди знали, хто і чим

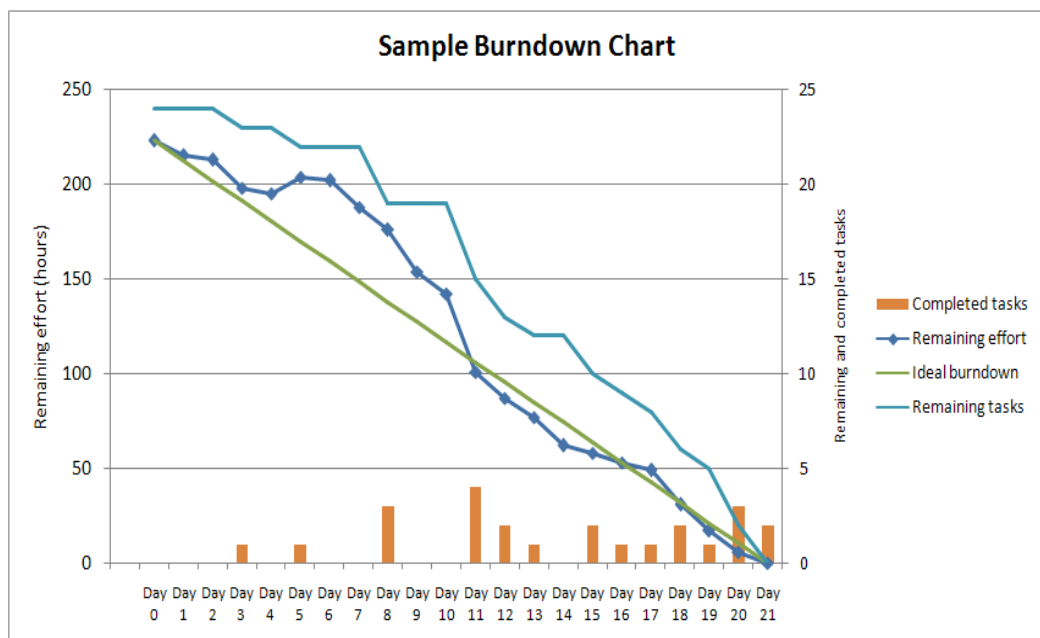
займається. Тривалість зустрічі строго обмежена і не повинна перевищувати 15 хвилин. Мета зустрічі – поділитися інформацією. Дана зустріч не призначена для розв'язання проблем, що виникли у проєкті. Усі спеціальні питання на цій зустрічі не розглядаються. Зустріч проводить Scrum-майстер. Він по колу задає питання кожному члену команди:

- 1) що зроблено вчора?
- 2) що буде зроблено сьогодні?
- 3) з якими проблемами зіткнувся?

Scrum-майстер збирає усі відкриті для обговорення питання у вигляді *Action Items*, наприклад у форматі що/хто/коли.

Щоденно оновлюється *Burndown chart* – діаграма, що ілюструє кількість роботи, що зроблена та залишилась. Існують різні види діаграми [10]:

- 1) діаграма спалювання робіт для спринту – показує кількість завдань, що зроблені, і кількість завдань, що треба зробити у наступному спринті;
- 2) діаграма спалювання робіт для випуску проєкту – показує кількість завдань, що зроблені, і кількість завдань, що треба зробити до випуску продукту (зазвичай будується на основі декількох спринтів).



Ретроспектива (Sprint Retrospective): наприкінці кожного спринту команда збирається на Ретроспективу, мета якої – переглянути якість існуючих процесів, взаємовідношення членів команди і інструменти, що застосовуються. Команда визначає, що зроблено добре, що зроблено не дуже

добре, а також виявляє потенційні можливості для поліпшень і складає план поліпшень на майбутнє.

Демонстрація робочої версії продукту Product Owner (Sprint Review): для обговорення, отримання вражень і побажань.

1.2 Визначення і структура бази даних

Визначення бази даних

1. Сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами, яка підтримує щонайменше одну з областей застосування.
2. У загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти.
3. Дані у базі організовують відповідно до моделі організації даних.

База даних – це організований набір даних, що зібрані у єдине ціле. Сенс БД полягає у зборі, збереженні, обробці і отриманні зв'язаних даних для використання додатком БД.

Додаток БД – це програма, що взаємодіє з БД для доступу і управління. БД зазвичай керує адміністратор БД (DBA).

Реляційна модель даних

1. Дані представлені у вигляді набору таблиць.
2. Кожний стовпець представляє атрибут (характеристику) таблиці.
3. Кожний рядок представляє екземпляр таблиці.
4. Кожна таблиця складається з рядків і стовпців.
5. Кожна таблиця має ідентифікатор (первинний ключ) – стовпець (поле) або набір стовпців, які однозначно ідентифікують рядок.
6. Порядок рядків і стовпців не має значення.
7. Кожний рядок є унікальним.
8. Кожний стовпець (поле) може містити тільки одне значення.
9. Значення у стовпці (полі) мають бути одного типу (*datatype*).
10. Імена таблиць мають бути унікальними.
11. Назви стовпців у кожній таблиці мають бути унікальними.

Реляційна база даних

1. Реляційна база даних (БД) зберігає інформацію у таблицях, що складаються зі стовпців і рядків.
2. Таблиця зберігає записи.
3. Рядки таблиці – це записи.
4. Стовпці таблиці – поля (атрибути).

Реляційна база даних – це набір записів, які зберігаються у таблицях. Кожна таблиця реляційної БД містить рядки записів і стовпці, що містять поля інформації (атрибути). Таблиця у реляційній БД також може називатися сутністю або об'єктом. Рядок у реляційній БД – екземпляр або запис. Кожна таблиця записів буде мати відношення з іншою таблицею записів, якщо вони використовують одне поле або атрибут (стовпець).

Таблиця – структура для організації і зберігання даних у реляційній БД:

Table: EMPLOYEES

| EMPLOYEE_ID | LAST_NAME | FIRST_NAME | DEPARTMENT_ID | PAYROLL_ID | NICKNAME |
|-------------|-----------|------------|---------------|------------|----------|
| 100 | SMITH | DANA | 10 | 21215 | Dana |
| 310 | ADAMS | TYLER | 15 | 59877 | Ty |
| 210 | CHEN | LAWRENCE | 10 | 1101 | Larry |
| 405 | GOMEZ | CARLOS | 10 | 52 | Chaz |
| 378 | LOUNGANI | NEIL | 22 | 90386 | Neil |

Primary Key Column (PK) Foreign Key Column (FK) Unique Key Column (UK)

Концептуальна модель БД

Концептуальна модель БД – систематизований змістовний опис модельованої БД на неформальній мові.

Проектування концептуальної моделі ґрунтується на аналізі завдань по обробці даних, що вирішуються в компанії. Концептуальна модель містить описи сутностей і їх взаємозв'язків, які представляють інтерес у предметній області, що розглядається, і отримані в результаті аналізу даних. Тобто дані, що вже використовуються, а також дані, що будуть реалізовані.

Проектування БД починається зовсім не з написання програми, а з аналізу процесу і технології роботи компанії.

Таким чином, *концептуальна модель*:

1. Відображає функціональні і інформаційні потреби бізнесу.
2. Основана на поточних потребах, але може відображати майбутні потреби.
3. Задовольняє потреби бізнесу (що ідеально теоретично), проте не відповідає їх реалізації (що фізично можливо).

Визначає:

1. Важливі сутності (стають таблицями у БД.)
2. Зв'язки між сутностями.

Не визначає:

1. Атрибути (стають полями і стовпцями у таблиці БД).
2. Унікальні ідентифікатори (атрибути, що стають первинними ключами у БД).

Логічна модель БД

Логічна модель БД описує дані більш докладніше (на відміну від концептуальної моделі), незалежно від того, як це буде реалізовано у фізичній моделі БД.

Метою побудови *логічної моделі БД* є отримання графічного представлення логічної структури досліджуваної предметної області. Логічна модель ілюструє сутності, а також взаємовідношення між ними. Сутності описують об'єкти, що є предметом діяльності предметної області, і суб'єкти, що здійснюють діяльність в рамках предметної області. Властивості об'єктів і суб'єктів реального світу описуються за допомогою атрибутів.

Взаємовідношення між сутностями ілюструються за допомогою зав'язків. Правила і обмеження взаємовідношень описуються за допомогою властивостей зав'язків. Зазвичай зв'язки визначають або залежності між сутностями, або вплив однієї сутності на іншу.

Основні вимоги до *вмісту логічної моделі*:

1. Логічна модель має відображати усі сутності і зв'язки системи.
2. Всі об'єкти моделі (і сутності, і зв'язки) мають бути іменовані (іменування сутностей і зав'язків має виконуватися в термінах предметної області).
3. Для зав'язків має бути вказана кратність (один-багато).

4. Для кожного зв'язку має бути вказаний напрям читання.
5. Для сутностей мають бути вказані основні атрибути (як мінімум).

Основні вимоги до *якості логічної моделі*:

1. Модель має читатися за схемою: <Сутність 1>-<відношення/вплив>-<Сутність 2>.
2. Модель має бути структурованою, сутності мають бути згруповані за логічним сенсом.
3. Бажано уникати перетину зав'язків.
4. Розташування об'єктів моделі має бути таким, щоб її зручно було читати.

Таким чином, *логічна модель*:

1. Містить усі сутності і зв'язки між ними.
2. Має назву – модель зв'язків між сутностями (ERM).
3. Ілюструється ERD-діаграмою.
4. Описує усі атрибути і унікальні ідентифікатори (UID, User identifier) для кожної сутності.
5. Визначає обов'язковість і необов'язковість атрибутів.
6. Визначає обов'язковість і необов'язковість зв'язків.

Фізична модель БД

Фізична модель БД – подання дизайну даних, як реалізованого чи призначеного для реалізації, у конкретній СУБД.

Фізичне моделювання перетворює логічну модель у модель реляційної БД. Кожна реляційна модель може мати одну або кілька фізичних моделей, по одній для кожної СУБД, яку Ви використовуєте.

Таким чином, *фізична модель БД*:

1. Є розширенням логічної моделі даних.
2. Вказує визначення таблиць, типи даних і точність.
3. Визначає представлення, індекси і інші об'єкти БД.
4. Описує, як мають бути реалізовані об'єкти у конкретній БД.
5. Демонструє усі структури таблиць – стовпці, первинні і зовнішні ключі.

Етапи створення *фізичної моделі даних*:

1 етап – Моделювання сутностей у вигляді таблиць.

2 етап – Моделювання зв'язків у вигляді зовнішніх ключів.

3 етап – Моделювання атрибутів у вигляді стовпців.

4 етап – Зміна моделі даних з урахуванням фізичних обмежень і вимог (модель необхідно протестувати, чи задовольняє таблиця 3 нормальній форм (3НФ). Як що ні, то треба провести процес нормалізації – провести покрокову декомпозицію (розбиття) таблиці на кілька таблиць).

Нормалізація БД

Нормалізація – це концепція реляційної БД, але її принципи можна застосувати до моделювання даних.

Метою є нормалізація даних до 3НФ (3NF) перед перетворенням моделі у реляційну модель.

| Правило | Опис |
|-----------------------------|---|
| Перша нормальна форма (1NF) | Усі атрибути повинні мати тільки одне значення |
| Друга нормальна форма (2NF) | Атрибут повинен залежати від повного UID його сутності |
| Третя нормальна форма (3NF) | Атрибут, який не є UID, не може залежати від іншого атрибуту, який не є UID |

Перша нормальна форма (1NF)

1. Для 1NF необхідна наявність атрибутів без декількох значень.
2. Щоб перевірити 1NF, перевірте, щоб кожний атрибут мав тільки одне значення для кожного екземпляру сутності.
3. Якщо атрибут має декілька значень, необхідно створити додаткову сутність і зв'язати її з початковою сутністю за допомогою зв'язку 1:М.

Друга нормальна форма (2NF)

1. 2NF вимагає, щоб будь-який атрибут, що не є UID, залежав (був властивістю або характеристикою) від усього UID.
2. Якщо UID є складеним, то кожний атрибут повинен залежати від усіх частин складеного UID.
3. Якщо атрибут не залежить від усього UID, необхідно створити додаткову сутність з частковим UID.

Третя нормальна форма (3NF)

1. Для 3NF атрибут, який не є UID (non-UID), не може залежати від іншого атрибуту, який також не є UID.
2. 3NF забороняє транзитивні залежності.
3. Транзитивна залежність існує у тому випадку, коли будь-який атрибут сутності залежить від іншого атрибуту сутності, який не є UID.
4. Необхідно перемістити у нову сутність усі non-UID атрибути, які залежать від інших non-UID атрибутів.

1.3 Тестування

Модульне тестування (Unit testing) – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми.

Модулем називають найменшу частину програми, яка може бути протестованою.

Check-list (контрольний список, що містить ряд необхідних перевірок для тестування) для тестування програмного продукту (Web-додаток) складається з:

- 1) тестування зручності використання;
- 2) функціонального тестування;
- 3) тестування сумісності;
- 4) тестування бази даних;
- 5) тестування безпеки;
- 6) тестування продуктивності.

Тестування зручності використання (юзабіліті) – це тестування доброзичливості додатку для користувача. При тестуванні зручності

використання перевіряється, чи легко новому користувачеві розібратися в додатку і тестується системна навігація.

Мета тестування зручності використання: впевнитися у простоті і ефективності використання продукту при використанні стандартних практик тестування зручності використання.

Сценарії тестування **зручності використання:**

- 1) вміст web-сторінки вірний, без граматичних і орфографічних помилок;
- 2) всі шрифти відповідають вимогам;
- 3) всі тексти правильно вирівняні;
- 4) всі повідомлення про помилки вірні, без орфографічних і граматичних помилок, і відповідають заголовку вікна;
- 5) підказки існують для всіх полів;
- 6) всі поля правильно вирівняні;
- 7) між полями, колонками, рядами і повідомленнями про помилки залишено досить вільного місця;
- 8) всі кнопки повинні мати стандартний формат і розмір;
- 9) посилання на домашню сторінку має бути на кожній сторінці сайту;
- 10) неактивні поля мають бути сірими;
- 11) перевірте, що на сайті немає битих посилань і зображень;
- 12) підтверджуючі повідомлення повинні відображатися для всіх операцій оновлення і видалення;
- 13) перевірте сайт при різних роздільних здатностях екрану (640 x 480, 600x800 і т. д.);
- 14) перевірте, що користувач може користуватися системою без роздратування;
- 15) перевірте, що *Tab* правильно працює;
- 16) панель скролу повинна з'являтися лише тоді, коли вона потрібна;
- 17) якщо при відправці форми є повідомлення про помилку, у ньому має міститися інформація, передана користувачем;
- 18) заголовок повинен відображатися на кожній сторінці;
- 19) всі поля (текстові, випадаючі меню, радіо-кнопки і т. д.) і кнопки мають бути доступні з клавіатури, і користувач має бути в змозі користуватися сайтом, використовуючи лише клавіатуру;
- 20) переконайтеся, що дані у випадаючих списках не обрізаються із-за розмірів поля, і перевірте, чи зашиті дані в код або керуються адміністратором.

Функціональне тестування – тестування функціональностей і операційної поведінки продукту з метою переконатися, що вони відповідають специфікаціям. Даний вид тестування ігнорує внутрішні механізми системи

або компоненту і концентрується виключно на вихідних даних, отриманих у відповідь на призначене для користувача введення і умови виконання сценаріїв.

Мета функціонального тестування: переконатися, що продукт відповідає потрібній функціональній специфікації, згаданій у документації з розробки.

Сценарії **функціонального тестування:**

- 1) протестуйте валідацію всіх обов'язкових полів;
- 2) переконайтеся, що знак зірочки відображається у всіх обов'язкових полях;
- 3) переконайтеся, що система не відображає вікно помилки при незаповнених необов'язкових полях;
- 4) переконайтеся, що коди коректно валідуються і не викликають помилок в розрахунках;
- 5) протестуйте числові поля: вони не повинні приймати літери, у випадку введення літери повинне відобразитися відповідне повідомлення про помилку;
- 6) протестуйте від'ємні значення в числових полях, якщо вони дозволені;
- 7) протестуйте, що ділення на нуль вірно обраховується;
- 8) протестуйте максимальну довжину кожного поля, щоб переконатися, що дані не обрізуються;
- 9) протестуйте спливаюче повідомлення («Це поле обмежене 500 знаками»), яке повинне відобразитися, якщо введені дані перевищують дозволений розмір поля;
- 10) переконайтеся, що підтверджуюче повідомлення відображається для операцій оновлення і видалення;
- 11) переконайтеся, що значення вартості відображуються в потрібній валюті;
- 12) протестуйте всі поля введення на спецсимволи;
- 13) протестуйте функціональність тайм-ауту;
- 14) протестуйте функціональність сортування;
- 15) протестуйте функціональність доступних кнопок;
- 16) протестуйте умови використання і питання, що часто ставляться: вони мають бути виразними і доступними користувачеві;
- 17) протестуйте, що при відмові функціональності користувач перенаправляється на спеціальну сторінку помилки;
- 18) протестуйте, що всі завантажені документи правильно відкриваються;
- 19) протестуйте, що користувач може скачати завантажені файли;
- 20) протестуйте поштову функціональність системи;

- 21) протестуйте, що Java Script вірно працює в різних браузерах (IE, Firefox, Chrome, Safari, Opera);
- 22) погляньте, що буде, якщо користувач видалить куки (cookies-файли), знаходячись на сайті;
- 23) погляньте, що буде, якщо користувач видалить куки після відвідування сайту;
- 24) протестуйте всі дані у випадючих списках: вони мають бути розташовані в хронологічному порядку.

Тестування сумісності перевіряє сумісність додатку з іншими елементами системи, в якій воно працює, – наприклад, браузерами, операційними системами або залізом.

Мета тестування сумісності: оцінка того, наскільки добре ПЗ працює в певному браузері, під певною ОС, з іншим ПЗ або залізом.

Сценарії *тестування сумісності:*

- 1) протестуйте сайт в різних браузерах (IE, Firefox, Chrome, Safari, Opera) і переконайтеся, що сайт правильно відображається;
- 2) переконайтеся, що використовувана версія HTML сумісна з відповідними версіями браузерів;
- 3) переконайтеся, що малюнки коректно відображаються в різних браузерах;
- 4) переконайтеся, що шрифти вірно відображуються в різних браузерах;
- 5) переконайтеся, що Java Script код працює в різних браузерах;
- 6) перевірте анімовані GIF в різних браузерах.

Інструмент для тестування сумісності *Spoon.net*: Spoon.net надає доступ до тисяч додатків (браузерів), що не вимагають установки. Цей інструмент допомагає тестувати додаток в різних браузерах на одній машині.

Тестування бази даних – перевіряються бекенд-записи, введені через web-додаток або десктоп-додаток. Дані, які відображаються у додатку, повинні збігатися з даними, що зберігаються в базі даних.

Що має знати тестувальник при тестування БД:

- 1) тестувальник має розуміти функціональні вимоги, бізнес-логіку, основний сценарій додатку і дизайн бази даних;
- 2) тестувальник має знатися на таблицях, тригерах, процедурах зберігання, способах відображення і покажчиках, використовуваних для додатку;

- 3) тестувальник має розуміти логіку тригерів, процедур зберігання, способів відображення і показчиків;
- 4) тестувальник має розуміти, які таблиці зачіпаються, коли операції вставки, оновлення і видалення виконуються в додатку.

Сценарії тестування *бази даних*:

- 1) перевірте назву бази даних: вона повинна збігатися із специфікацією;
- 2) перевірте таблиці, колонки, типи колонок і значення за умовчанням: все це має збігатися із специфікацією;
- 3) перевірте, чи дозволяє колонка значення *null*;
- 4) перевірте первинний і зовнішній ключ кожної таблиці;
- 5) перевірте процедури зберігання;
- 6) протестуйте, чи встановлена процедура зберігання;
- 7) перевірте назву процедури зберігання;
- 8) перевірте назви параметрів, їх типи і кількість;
- 9) перевірте, обов'язкові параметри чи ні;
- 10) перевірте процедуру зберігання, видаливши деякі параметри;
- 11) перевірте базу даних, якщо на виході нуль – записи з нулем мають бути задіяні;
- 12) перевірте процедуру зберігання, задавши прості SQL-запити;
- 13) переконайтеся, що процедура повертає значення;
- 14) перевірте процедуру введенням тестових даних;
- 15) перевірте поведінку кожного прапора у таблиці;
- 16) переконайтеся, що дані правильно зберігаються в базі даних після кожного введення;
- 17) перевірте дані під час кожної операції оновлення, видалення і вставки;
- 18) перевірте довжину кожного поля: довжина на бекенді і фронтенді мають збігатися;
- 19) перевірте назви баз даних QA, UAT і прод, імена мають бути унікальними;
- 20) перевірте зашифровані дані в БД;
- 21) перевірте розмір БД і час відгуку на кожен запит;
- 22) перевірте дані, що відображаються на фронтенді, і переконайтеся, що вони збігаються з бекендом;
- 23) перевірте цілісність даних, вводячи невалідні значення в БД;
- 24) перевірте тригери.

Тестування безпеки націлено на пошук недоліків і пропусків з точки зору безпеки додатку.

Сценарії *тестування безпеки*:

- 1) переконайтеся, що сторінки, що містять важливі дані (пароль, номер кредитної картки, відповіді на секретні питання і т. п.) відкриваються через HTTPS (SSL);
- 2) переконайтеся, що важлива інформація (пароль, номер кредитної картки) відображається у зашифрованому вигляді;
- 3) переконайтеся, що правила створення паролів упроваджені на всіх сторінках авторизації (реєстрація, сторінка «забули пароль», зміна паролю);
- 4) переконайтеся, що якщо пароль змінений, користувач не може зайти під старим паролем;
- 5) переконайтеся, що повідомлення про помилки не містять жодної секретної інформації;
- 6) переконайтеся, що якщо користувач вийшов з системи або сесія завершена, він не може користуватися сайтом;
- 7) перевірте доступ до закритих і відкритих сторінок сайту безпосередньо без авторизації;
- 8) переконайтеся, що опція «Перегляд початкового коду» відключена і її не бачить користувач;
- 9) переконайтеся, що обліковий запис користувача блокується, якщо він кілька разів ввів пароль невірно;
- 10) переконайтеся, що пароль не зберігається в кукі;
- 11) переконайтеся, що якщо яка-небудь функціональність не працює, система не відображає інформацію про додаток, сервер або базу даних; замість цього відображається відповідне повідомлення про помилку;
- 12) перевірте сайт на SQL-ін'єкції;
- 13) перевірте права користувачів і їх ролі (наприклад, кандидат не має бути здатний дістати доступ до сторінки адміністратора);
- 14) переконайтеся, що важливі операції пишуться в логи і інформацію можна відстежити;
- 15) переконайтеся, що значення сесій відображаються в адресному рядку у зашифрованому вигляді;
- 16) переконайтеся, що кукі зберігаються в зашифрованому вигляді;
- 17) перевірте додаток на стійкість до брутфорс-атак (брутфорс атака (brute force) – метод підбору паролю, шляхом почергового перебору можливих комбінацій. Зазвичай брутфорс виробляється масово з використанням стандартних комбінацій логінів (admin, administrator) і паролів (осмислені словосполучення, словники)).

Тестування продуктивності – проводиться для оцінки відповідності системи або компоненту специфічним вимогам до продуктивності.

Сценарії *тестування продуктивності*:

- 1) визначення продуктивності, стабільності і масштабованості додатку під різним навантаженням;
- 2) визначення, чи може актуальна архітектура підтримувати додаток при пікових навантаженнях;
- 3) визначення, яка конфігурація призводить до найкращих показників продуктивності;
- 4) визначення, чи не змінився час відгуку в новій версії додатку;
- 5) визначення пляшкового горла додатку і інфраструктури;
- 6) оцінка продукту і заліза з метою упевнитися, що вони витримають прогнозовані обсяги навантаження.

Неможливо провести *тестування продуктивності вручну* з ряду причин:

- 1) знадобиться велика кількість ресурсів;
- 2) неможливо одночасно здійснювати ряд дій;
- 3) відсутній відповідний спосіб відстежування поведінки системи;
- 4) складність виконання завдань, що повторюються.

Щоб впоратися з вище переліченими проблемами, використовуються спеціальні інструменти тестування продуктивності:

- 1) Apache Jmeter;
- 2) Load Runner;
- 3) Borland Silk Performer;
- 4) Rational Performance Tester;
- 5) WAPT;
- 6) NEO LOAD.

2 СТРУКТУРА ПРОЄКТУ

| | |
|---|--|
| Обсяг | 240 годин / 8 кредитів ECTS |
| Виконання | командне (можливе індивідуальне виконання при попередньому погодженні з Керівником Проєкту) |
| Управління | <i>Scrum</i> -методологія |
| Команда | <ul style="list-style-type: none"> - Власник Продукту (<i>Product Owner</i>), Керівник (<i>Scrum Master</i>), Команда розробників (<i>Scrum Team</i>): від 3 до 5 здобувачів - роль <i>Product Owner</i> виконує Керівник Проєкту - роль <i>Scrum Master</i> поділяють Керівник Проєкту і Капітан Команди |
| Кількість спринтів | 12 |
| Обов'язкові церемонії | <ul style="list-style-type: none"> - планування спринту (обов'язково за участю <i>Product Owner</i>) - щоденні наради - демонстрації - ретроспектива (обов'язково за участю <i>Product Owner</i>) |
| Обов'язкові складові результату виконання проєкту | <ul style="list-style-type: none"> - реліз програмного забезпечення - пояснювальна записка - презентація продукту + текст доповіді - спланований сценарій демонстрації роботи програмного забезпечення |

3 ПОРЯДОК ВИКОНАННЯ ПРОЄКТУ

3.1 Етапи виконання Проєкту

| № | Спринт | Дедлайн | Результат |
|----|---------------------------------------|---------------|---|
| 1 | Формування проєктної команди | до 12.01.202X | Затверджений склад проєктної команди |
| 2 | Дослідження ринку | до 18.01.202X | Документ з пропозиціями щодо нового продукту |
| 3 | Аналіз вимог | до 22.01.202X | Документ з вимогами до нового продукту |
| 4 | Проектування взаємодії з користувачем | до 03.05.202X | Візуалізовані приклади використання, спрощені та деталізовані прототипи продукту |
| 5 | Візуальний дизайн | до 03.05.202X | Графічні файли з кінцевим дизайном продукту |
| 6 | Проектування бази даних | до 11.05.202X | ER-діаграми рівня концептуальної, логічної та фізичної моделей даних |
| 7 | Проектування програмного забезпечення | до 18.05.202X | Файли з діаграмами та схемами компонент програмного забезпечення |
| 8 | Розробка програмного забезпечення | до 21.05.202X | Розроблений програмний продукт |
| 9 | Тестування програмного забезпечення | до 25.05.202X | Перелік помилок до виправлення |
| 10 | Підготовка пояснювальної записки | до 28.05.202X | Пояснювальна записка |
| 11 | Підготовка презентації | до 31.05.202X | Презентація продукту та спланований сценарій демонстрації роботи програмного забезпечення |
| 12 | Захист | до 04.06.202X | Підсумкова оцінка за виконання Проєкту |

3.2 Реліз програмного забезпечення

Реліз програмного забезпечення повинен відповідати перерахованим нижче вимогам:

- 1) мати базові програмні компоненти: реляційну базу даних, рівень доступу до даних, рівень програми та рівень інтерфейсу;
- 2) мати часткове покриття *unit*-тестами;
- 3) вихідний код програми відповідати стандарту оформлення коду;
- 4) мати відповідні коментарі або *doc-strings* до програмних елементів;
- 5) проєкт розміщено в репозиторії на GitHub.

3.3 Структура та вимоги до пояснювальної записки

Під час підготовки пояснювальної записки треба дотримуватися основних вимог до її змісту.

Пояснювальна записка умовно поділяється на:

- 1) вступну частину;
- 2) основну частину;
- 3) додатки.

Вступна частина містить такі структурні елементи:

- 1) титульний аркуш;
- 2) реферат.

Основна частина містить такі структурні елементи:

- 1) вступ;
- 2) змістовну частину звіту (суть пояснювальної записки);
- 3) висновки;
- 4) перелік джерел посилання.

Рекомендований обсяг пояснювальної записки не повинен перевищувати 50 сторінок.

Короткі настанови до обов'язкових структурних елементів пояснювальної записки наведені у таблиці 1.

Таблиця 1 – Настанови до обов'язкових структурних елементів пояснювальної записки до Проєкту третього рівня

| Назва розділу | Назва підрозділу | Вимоги та обов'язкові елементи |
|------------------------------------|------------------|---|
| Титульний лист | - | Див. у Додаток А |
| Лист завдання | - | - |
| Календарний план виконання проєкту | - | - |
| Реферат | - | - не більше 1 сторінки - українською та англійською мовами |

| | | |
|--|--|---|
| Зміст | - | - |
| Вступ | - | <ul style="list-style-type: none"> - актуальність теми - мета і задачі проєкту - практична цінність проєкту - структура та обсяг пояснювальної записки |
| Аналіз предметної області і постановка задачі | Проблеми розвитку інформаційних технологій | <ul style="list-style-type: none"> - відомі теоретичні і практичні результати, які отримані іншими вітчизняними та іноземними авторами за напрямом роботи, із посиланнями на їх праці |
| | Аналіз існуючих інформаційних технологій | <ul style="list-style-type: none"> - можливості, переваги і недоліки відомих інформаційних технологій, які в повній мірі або частково вирішують задачі, які будуть закладені у новий продукт |
| | Висновки за розділом | <ul style="list-style-type: none"> - результати аналізу предметної області - постановка задачі |
| Проектування застосунку | Аналіз варіантів використання | <ul style="list-style-type: none"> - функціональні та нефункціональні вимоги до нового продукту - діаграма прецедентів (<i>Use case diagram</i>) |
| | Моделювання процесів | <ul style="list-style-type: none"> - діаграми діяльності (<i>Activity diagram</i>) - математична модель процесів (за наявності) |
| | Моделювання даних | <ul style="list-style-type: none"> - ідентифіковані сутності, атрибути сутностей та відношення між сутностями - ER-діаграма (<i>Entity relationship diagram</i>) - нормалізація ER-діаграми (приведення до третьої нормальної форми) |
| | Проектування архітектури | <ul style="list-style-type: none"> - ідентифіковані програмні (наприклад, сховище даних, рівень доступу до даних, рівень програми та рівень інтерфейсу) та апаратні (наприклад, клієнтські комп'ютери, сервери та мережеве середовище) компоненти - схема розподілу програмних компонентів між апаратними компонентами - вимоги: експлуатаційні, до виконання, безпеки, культурні та політичні |
| | Проектування структури | <ul style="list-style-type: none"> - UML-діаграми рівня проектування програмного забезпечення |
| | Висновки за розділом | <ul style="list-style-type: none"> - результати етапу проектування |
| | Реалізація застосунку | Особливості реалізації |
| Конструювання | | <ul style="list-style-type: none"> - діаграма компонентів (<i>Component diagram</i>) |

| | | |
|------------------------|-----------------------|---|
| | | - діаграма пакетів (<i>Package diagram</i>) |
| | Реалізація бази даних | <ul style="list-style-type: none"> - опис таблиць (назва, поля із типами даних, ключі, обмеження) - повна схема даних у вибраній системі управління базами даних - опис SQL-запитів до бази даних - опис процедур, функцій та тригерів до бази даних (за наявності) - механізми захисту даних (за наявності) |
| | Документування | <ul style="list-style-type: none"> - схема організації довідкових технічних документів - схема організації користувацьких документів |
| | Адміністрування | <ul style="list-style-type: none"> - інструкції щодо інсталяції програмного забезпечення - інструкції щодо конфігурації елементів програмного забезпечення - інструкції щодо адміністрування даних |
| | Тестування | <ul style="list-style-type: none"> - модульне тестування - функціональне тестування |
| | Висновки за розділом | <ul style="list-style-type: none"> - результати етапу реалізації |
| Висновки | - | <ul style="list-style-type: none"> - стислий виклад результатів вирішення поставленого завдання, що отримані в процесі аналізу предметної області, проектування та реалізації програмного забезпечення - відомості про достовірність отриманих результатів - не більше 1 сторінки |
| Список посилань | - | <ul style="list-style-type: none"> - перелік використаних джерел у порядку згадування по тексту мовою оригіналу джерел відповідно до вимог ДСТУ 8302:2015 - не менше 20 найменувань |
| Додатки | - | <ul style="list-style-type: none"> - екрани застосунку - фрагменти лістингу - SQL-запити до бази даних |

Вимоги до оформлення пояснювальної записки:

1. Формат аркушу - А4.
2. Обсяг основного тексту пояснювальної записки не повинен перевищувати 50 сторінок; загальний обсяг не повинен перевищувати 70 сторінок.

3. Оформлення пояснювальної записки повинно відповідати вимогам стандарту ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання» [4].
4. Оформлення списку посилань повинно відповідати вимогам стандарту ДСТУ 8302:2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання» [5].
5. Вимоги до оформлення основного тексту:
 - a. параметри сторінки (поля): ліве – 2,5 см, праве – 1,5 см, верхнє і нижнє – 2 см;
 - b. основний шрифт – Times New Roman, 14 пт;
 - c. міжрядковий інтервал – 1,5;
 - d. інтервал до та після рядків – 0;
 - e. абзацний відступ – 1,25 см;
 - f. вирівнювання тексту - по ширині.
6. Кожний розділ починається з нової сторінки.
7. Назва розділу оформлюється у верхньому регістрі.
8. Номер рисунків та таблиць складається з двох цифр, перша – номер розділу, в якому розташований рисунок або таблиця, друга – порядковий номер рисунку або таблиці у розділі.
9. Рисунки та таблиці розміщуються після першого посилання на них у тексті (рис. 1.1 або табл. 1.1). Після підпису рисунку або таблиці обов'язково наводяться дані про джерело, звідки вони взяті (розроблено автором (-ами) / розроблено автором (-ами) за даними [номер посилання із зазначенням номеру сторінки] / [номер посилання із зазначенням номеру сторінки]).
10. Нумерація сторінок – внизу по центру.
11. Вимоги до оформлення тексту додатків:
 - a. параметри сторінки (поля): ліве – 2,5 см, праве – 1,5 см, верхнє і нижнє – 2 см;
 - b. основний шрифт – Courier New, 12 пт;
 - c. міжрядковий інтервал – 1;
 - d. інтервал до та після рядків – 0;
 - e. абзацний відступ – 0;
 - f. вирівнювання тексту – по лівому краю.

3.4 Структура та вимоги до презентації

Презентація має містити наступні слайди:

- 1) титульний слайд;
- 2) аналіз предметної області;

- 3) постановка задачі;
- 4) аналіз варіантів використання;
- 5) діаграми діяльності;
- 6) концептуальна модель даних;
- 7) логічна модель даних;
- 8) архітектура;
- 9) візуальний дизайн;
- 10) дизайн;
- 11) засоби реалізації;
- 12) фізична модель даних;
- 13) SQL-запити;
- 14) діаграми компонентів і пакетів;
- 15) екрани застосунку;
- 16) висновки.

Короткі настанови до обов'язкових структурних елементів презентації наведені у таблиці 2.

Таблиця 2 – Настави до обов'язкових структурних елементів презентації до Проекту третього рівня

| Назва слайду (ів) | Вимоги та обов'язкові елементи |
|-------------------------------|---|
| Титульний слайд | <ul style="list-style-type: none"> - назва продукту - члени команди із зазначення ролей |
| Аналіз предметної області | <ul style="list-style-type: none"> - актуальність теми - мета і задачі проєкту |
| Постановка задачі | <ul style="list-style-type: none"> - стисло перелік основних задач - опис вхідних і вихідних даних |
| Аналіз варіантів використання | <ul style="list-style-type: none"> - список функціональних та нефункціональних вимоги до нового продукту - діаграма прецедентів (<i>Use case diagram</i>) |
| Діаграми діяльності | <ul style="list-style-type: none"> - діаграма діяльності (<i>Activity diagram</i>) |
| Концептуальна модель даних | <ul style="list-style-type: none"> - діаграми відношень (<i>Entity relationship diagram</i>) |
| Логічна модель даних | <ul style="list-style-type: none"> - діаграми відношень (<i>Entity relationship diagram</i>) |
| Архітектура | <ul style="list-style-type: none"> - схема розподілу програмних компонентів між апаратними компонентами |
| Візуальний дизайн | <ul style="list-style-type: none"> - діаграма навігації по екранах (<i>User flow diagram</i>) - діаграма структури інтерфейсу (<i>Interface structure diagram</i>) - прототип інтерфейсу (<i>UI prototype</i>) |
| Дизайн | <ul style="list-style-type: none"> - UML-діаграми рівня проєктування програмного забезпечення |
| Засоби реалізації | <ul style="list-style-type: none"> - перелік засобів реалізації програмного забезпечення - перелік засобів управління проєктом |

| | |
|--------------------------------|---|
| Фізична модель даних | - повна схема даних у вибраній системі управління базами даних |
| SQL-запити | - перелік реалізованих SQL-запитів до бази даних |
| Діаграми компонентів і пакетів | - діаграма компонентів (<i>Component diagram</i>) - діаграма пакетів (<i>Package diagram</i>) |
| Екрани застосунку | - головне вікно застосунку - дочірні вікна застосунку |
| Висновки | - найбільш важливі практичні результати, одержані під час виконання проєкту - якісні та кількісні показники здобутих результатів |

Вимоги до оформлення презентації:

1. Для підготовки презентації рекомендовано використовувати корпоративний шаблон презентації, який доступний за адресою <https://www.krok.edu.ua/download/brand-book/krok-template-presentation.pptx>.
2. Всі слайди (крім першого) повинні містити порядковий номер, розташований у правому нижньому кутку.
3. Кожен слайд (крім першого) повинен мати коротку назву (заголовок, без крапки в кінці).
4. Вимоги до оформлення тексту презентації:
 - a. шрифт заголовків – Calibri, 36 пт, напівжирний;
 - b. шрифт порядкових номерів слайдів - Calibri, 20 пт, напівжирний;
 - c. шрифт основного тексту - Calibri, 22 пт;
 - d. міжрядковий інтервал – 1;
 - e. інтервал до та після рядків – 0;
 - f. абзацний відступ – 0 см;
 - g. вирівнювання тексту - по лівому краю.

4 КРИТЕРІЇ ОЦІНЮВАННЯ ПРОЄКТУ

Підсумкове оцінювання за виконання Проєкту, яке відображає рівень досягнення запланованих результатів навчання, проводиться за 100-бальною накопичувальною шкалою.

Підсумкова оцінка розраховується за формулою:

$$G = 0,5 \cdot G_{team} + 0,25 \cdot G_{report} + 0,15 \cdot G_{presentation} + 0,1 \cdot G_{oracle_test},$$

де

G_{team} – середня оцінка членів команди, у т.ч. Product Owner,

G_{report} – оцінка за пояснювальну записку,

$G_{presentation}$ – оцінка за презентацію продукту,

G_{oracle_test} – оцінка за сертифікацію з курсу Database Foundations від Oracle Academy.

Для оцінювання рівня з G_{team} , G_{report} , та $G_{presentation}$ використовується 12-бальна оцінна шкала з подальшим переведенням у 100-бальну шкалу:

| Рівні оцінної шкали | Відмітки | Інтерпретація рівня досягнення результатів навчання курсу |
|---------------------|----------|---|
| IV високий | 12 | чудово |
| | 11* | відмінно |
| | 10 | майже відмінно |
| III середній | 9 | більш ніж добре |
| | 8* | добре |
| | 7 | майже добре |
| II достатній | 6 | більш ніж задовільно |
| | 5* | задовільно |
| | 4 | майже задовільно |
| I початковий | 3 | мало задовільно |
| | 2* | незадовільно |
| | 1 | знання майже відсутні |

* – базова оцінка рівня оцінної шкали

5 ДОТРИМАННЯ ВИМОГ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ПРИ ВИКОНАННІ ПРОЄКТУ

До початку виконання Проєкту необхідно уважно ознайомитися з Положенням про академічну доброчесність в Університеті КРОК [2], Положенням про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» [3], Рекомендаціями МОН України щодо запобігання академічному плагіату та його виявлення в наукових роботах (авторефератах, дисертаціях, монографіях, наукових доповідях, статтях тощо) [9].

Основні види порушень академічної доброчесності, яких слід уникати під час виконання Проєкту, написання пояснювальної записки, підготовки презентації [2]:

плагіат – оприлюднення (опублікування), повністю або частково, чужого твору під іменем особи, яка не є автором цього твору (пункт в ст. 50 Закону України «Про авторське право і суміжні права»);

академічний плагіат – оприлюднення (частково або повністю) наукових (творчих) результатів, отриманих іншими особами, як результатів власного дослідження (творчості) та/або відтворення опублікованих текстів (оприлюднених творів мистецтва) інших авторів без зазначення авторства (ч. 4 ст. 42 Закону України «Про освіту»);

самоплагіат – оприлюднення (частково або повністю) власних раніше опублікованих наукових результатів як нових наукових результатів;

фабрикація – вигадкування даних чи фактів, що використовуються в освітньому процесі або наукових дослідженнях;

фальсифікація – свідомо зміна чи модифікація вже наявних даних, що стосуються освітнього процесу чи наукових досліджень.

При використанні фрагментів, запозичених з інших джерел, необхідно прийняти до уваги, що вони будуть **ідентифіковані як плагіат** за умови, якщо будуть відтворені у тексті роботи [9]:

- 1) без змін, з незначними змінами, або в перекладі, обсягом від речення і більше, без посилання на автора (авторів) відтвореного тексту;
- 2) повністю або частково, через перефразування чи довільний переказ без посилання на автора (авторів) відтвореного тексту;
- 3) як цитати з третіх джерел без вказування, за яким саме безпосереднім джерелом наведена цитата;
- 4) як наведена в іншому джерелі науково-технічна інформація (крім загальновідомої) без вказування на те, з якого джерела взята ця інформація;

- 5) як оприлюднені твори мистецтва без зазначення авторства цих творів мистецтва.

У контексті технічної (на текстові збіги) та експертної перевірки (керівниками) повного тексту роботи плагіатом буде вважатись наступне [3]:

- 1) видання виконаної іншим автором роботи за свою без внесення в неї жодних змін;
- 2) дослівне копіювання фрагментів тексту (від фрази до набору речень) без належного оформлення цитування;
- 3) внесення незначних правок у скопійований матеріал (переформулювання речень, зміна порядку слів у них тощо) та без належного оформлення цитування;
- 4) представлення суміші власних і запозичених аргументів без належного цитування;
- 5) *парафраза* – переказ своїми словами чужих думок, ідей або тексту; сутність парафрази полягає в заміні слів (знаків), фразеологічних зворотів або пропозицій при використанні будь-якої авторської наукової праці (збереженої на електронних або паперових носіях, у тому числі розміщеної в мережі Інтернет);
- 6) *компіляція* – створення значного масиву тексту без поглибленого вивчення проблеми шляхом копіювання тексту із низки джерел без внесення в нього правок, з посиланням на авторів та «маскуванням» шляхом написання перехідних речень між скопійованими частинами тексту.

Положенням про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» передбачено виявлення рівня оригінальності тексту представленої роботи у відсотках, визначених за допомогою програмно-технічних засобів перевірки на текстові збіги. За результатами такої перевірки передбачені наступні дії:

- 1) при виявленні високого рівня оригінальності роботи (понад 75%) текст вважається оригінальним, додаткові дії щодо запобігання неправомірним запозиченням не проводитимуться;
- 2) при виявленні задовільного рівня оригінальності роботи (55-75%) вважається, що в тексті роботи наявні окремі ознаки академічного плагіату. Додатково електронний звіт з результатами антиплагіатної перевірки з гіперпосиланнями на запозичені джерела буде експертно перевірений (науковим керівником) на наявність посилань на першоджерела для цитованих фрагментів, в разі необхідності буде проведена повторна технічна перевірка;

- 3) при виявленні низького рівня оригінальності твору (35-55%) вважається, що в тексті роботи наявні певні ознаки академічного плагіату, проте робота може бути прийнята за умови доопрацювання з обов'язковою наступною перевіркою на оригінальність;
- 4) при виявленні рівня оригінальності роботи нижче за 35%, вважається, що в тексті роботи наявні істотні ознаки плагіату. Робота до розгляду (захисту) не приймається.

У разі виявлення академічного плагіату у Проєкті, він знімається із захисту без права повторного захисту.

Відсоток рівня оригінальності роботи може бути збільшений за рахунок правомірних запозичень, до яких належать:

- 1) власні назви (індивідуальні найменування окремих одиничних об'єктів, у т.ч. найменування установ, назви праць, які досліджувалися в роботі, бібліографічні посилання на джерела та ін.);
- 2) усталені словосполучення, що характерні для сфери ІТ, комп'ютерних наук;
- 3) належним чином оформлені цитування;
- 4) самоцитування (фрагменти текстів, що належать автору роботи, опубліковані або оприлюднені в електронній формі ним у інших творах).

Під час виконання Проєкту та написання пояснювальної записки, необхідно переконатися, що посилання у тексті пояснювальної записки зроблені правильно.

Посилання у тексті робиться при цитуванні джерела чи думки дослідника, при вказівці на певне важливе свідчення джерела, при запозиченні положень, використанні фактичного матеріалу, результатів досліджень інших авторів, посилань на досвід.

Кожну цитату обов'язково супроводжувати посиланням на джерело у тексті. Позначається таке посилання порядковим номером за переліком джерел посилання у клямрах (наприклад, «... у працях [1–7]...»).

Якщо використовуються відомості, матеріали з монографій, оглядових статей, інших джерел з великою кількістю сторінок, тоді у посиланні необхідно точно вказати номери сторінок, ілюстрацій, таблиць, формул з джерела, на які є посилання (наприклад, [3, с. 29]).

Якщо текст цитується не за першоджерелом, а за іншим виданням чи документом, то необхідно розпочинати посилання словами: «Цит. за:».

При непрямому цитуванні (переказі, викладенні думок інших авторів своїми словами), що дає значну економію тексту, треба бути гранично точними, коректними.

Необхідно уникати як надмірного, так і недостатнього цитування, оскільки це знижує рівень: надмірне цитування створює враження компілятивності праці, а недостатнє – знижує цінність викладеного матеріалу.

Якщо наводиться цитата і треба виділити у ній деякі слова, необхідно зробити спеціальний припис (після тексту, який пояснює виділення, треба поставити крапку, далі дефіс і вказати ініціали автора роботи, а весь текст припису помістити у дужки) (наприклад, (курсив наш – А.А.), (підкреслено мною – А.А.), (доповнено мною – А.А.)).

Бібліографічний опис використаних джерел оформлюйте відповідно до вимог ДСТУ 8302:2015 [5].

Відповідальність за перевірку Проєкту у встановлені терміни, прийняття рішення щодо доопрацювання та повторну перевірку на плагіат, про допуск до захисту несе Керівник Проєкту (Product Owner).

6 ТЕХНІЧНЕ ОБЛАДНАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Технічне обладнання: комп'ютер/ноутбук.

Програмне забезпечення:

- 1) *Data Modeling with Oracle SQL Developer*
(<https://www.oracle.com/database/technologies/appdev/datamodeler.html>);
- 2) інтегроване середовище розробки (обирається командою);
- 3) система управління базою даних (обирається командою);
- 4) VCS GIT (<https://git-scm.com/>).

7 ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про організацію освітнього процесу в Університеті «КРОК». URL:https://library.krok.edu.ua/media/library/category/publicna-informatsiya/3-3_osvitnij-protses-2020-11-02.pdf
2. Положення про академічну доброчесність в Університеті «КРОК». URL:https://www.krok.edu.ua/download/nakazi/2018-10-18_kodeks-akademichnoi-dobrochesnosti.pdf
3. Положення про перевірку академічних та наукових текстів на плагіат в Університеті «КРОК» URL:https://library.krok.edu.ua/media/library/category/publicna-informatsiya/2020-04-10_pozhennya-pro-perevirku-tekstiv-na-plagiat.pdf
4. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. URL:http://www.knmu.kharkov.ua/attachments/3659_3008-2015.pdf
5. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання. URL:<http://pdf.lib.vntu.edu.ua/books/2018/%D0%94%D0%A1%D0%A2%D0%A3%208302%20%D0%BF%D0%BE%D0%B2%D0%BD%D0%B8%D0%B9.pdf>
6. Освітньо-професійна програма «Комп'ютерні науки» Університету «КРОК». URL:<https://library.krok.edu.ua/media/library/category/litsenzuvannia-ta-akredytatsiia/osvitni-prohramy/komp-yuterni-nauki-bakalavr-2021-04-29.pdf>
7. Стандарт вищої освіти першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп'ютерні науки». URL:<https://mon.gov.ua/storage/app/media/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyuterni-nauki-bakalavr.pdf>
8. Положення про політику та процедури врегулювання конфліктних ситуацій в Університеті «КРОК». URL:https://library.krok.edu.ua/media/library/category/publicna-informatsiya/2021-07-07_pozhennya-pro-protseduri-vregulyuvannya-konfliktnikh-situatsij.pdf
9. Рекомендації МОН України щодо запобігання академічному плагіату та його виявлення в наукових роботах (авторефератах, дисертаціях, монографіях, наукових доповідях, статтях тощо). URL:<https://zakon.rada.gov.ua/rada/show/v8681729-18#Text>
10. Burndown chart. URL: https://en.wikipedia.org/wiki/Burndown_chart
11. INTRO ON AGILE SCRUM. URL: <https://agileevangelists.wordpress.com/2014/01/23/intro-on-agile-scrum/>

ДОДАТОК А
Форма титульного аркушу пояснювальної записки Проєкту

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ПРОЄКТ ТРЕТЬОГО РІВНЯ

Тема: «.....»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав (-ла): здобувач (ка) 3 курсу
групи

Прізвище, ініціали

Керівник: *науковий ступінь, посада, ПІБ керівника*